LINUBIA: A Linux-Supported User-Based IP Accounting

Cristian Morariu, Manuel Feier, and Burkhard Stiller

Department of Informatics IFI, University of Zurich, Switzerland
{morariu,stiller}@ifi.uzh.ch, manuel@feier.ch

Abstract. Obtaining information about the usage of network ressources by individual users forms the basis for establishing network billing systems or network management operations. While there are already widely used accounting techniques available for measuring IP network traffic on a per-host basis, there is no adequate solution for accounting per-user network activities on a multiuser operating system. This work provides a survey on existing approaches to this problem and identifies requirements for a user-based IP accounting module. It develops a suitable software architecture LINUBIA and proposes a prototypical implementation for the Linux 2.6 operating system, which is capable of providing per-user accounting for both the IPv4 and the IPv6 protocol.

1 Introduction and Problem Statement

The Internet is rapidly growing and fundamentally influencing economic, cultural, and social developments worldwide. While more and more people take advantage of this global network and new services are being deployed every day, more network resources are consumed. This creates the need for effective accounting mechanisms that are closely coupled to authentication mechanisms, *e.g.,* in support of network management tasks, charging requirements, or intrusion detection systems for systems and users. Often it becomes necessary to know what amount and which type of network traffic a specific network user is generating.

Today, as networking is moving towards an all-IP network [9] an accounting system integrated into the IP layer seems the most straight forward solution. This approach allows for the same accounting mechanisms to be used regardless of the application and the transport protocol carried over IP, or the data link layer and physical connection the IP runs on top of.

Although the accounting of IP network traffic has received wide attention since the beginning of the Internet [18], existing systems have a major drawback by looking strictly to the IP packet captured on the wire. Such an approach allows for the mapping of each IP packet to an end-system, which sends or receives the packet, but it is unable to specify, which user was responsible for generating the traffic. Multiuser operating systems often use a single IP address, which is shared among different individual users. Since multiple users may be connected remotely at the same time to the same machine and may have different applications that generate IP traffic being transported over the

A. Clemm, L.Z. Granville, and R. Stadler (Eds.): DSOM 2007, LNCS 4785, pp. 229–241, 2007.

network, it is impossible to identify how much traffic each of theses users generated by just looking into the IP traffic at the router level.

Therefore, this paper proposes the user-based IP accounting architecture named LINUBIA, which uses a Linux kernel extension and a library for accessing this extension, for mapping each IP packet sent or received to the responsible user. This solution allows for splitting network costs in case of usage-based charging or may allow detection of the user or process that was responsible for illegal IP traffic.

Fig. 1 depicts a typical scenario for using the new user-based IP accounting infrastructure. In an enterprise network users are typically authenticated by using a centralized authentication server such as LDAP (Light-weight Directory Access Protocol) [15] or Kerberos [12] and they may access the network from any terminal or working station that is configured to use the central authentication server. Upon authentication the device to which the user logged on to starts to meter the network usage and sends periodic accounting records to the accounting server. Since the network usage is mapped to user identifiers (ID) and a user uses the same ID with any device he is allowed to connect to, the accounting server may aggregate the network usage from different devices within the network and present users with detailed and aggregated information about network traffic they created.

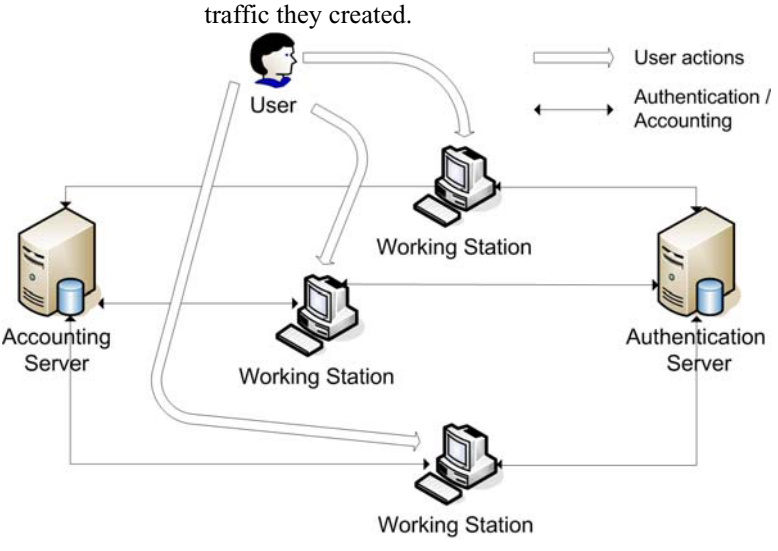

Fig. 1. Enterprise Scenario

The remainder of the paper is structured as follows. Section 2 presents a brief overview on related work in the field of IP accounting followed by Section 3, which outlines the design of the new approach. Section 4 gives major implementation details. While Section 5 discusses evaluation results, finally, Section 6 draws conclusions and provides ideas about future work.

2 Related Work

Some of the first guidelines for measuring internet traffic have been proposed in [18]. The IETF defined a generic AAA architecture (Authentication, Authorization, and Accounting) [10] that formed the basis for RADIUS [14] and Diameter [2] protocols. The AAA architecture and its related protocols do not define how measurements need to be done, but specify how to transport and collect the data measured. SNMP (Simple Network Management Protocol) [3] allows network administrators to monitor and manage network traffic performance. Each network device in a SNMP managed network has a management agent that resides on that device. The management agent collects information from different counters and configuration parameters of the managed device and makes them available to a Network Monitoring Station (NMS) via the SNMP protocol. The information collected via SNMP is typically highly aggregated (e.g., summary of data transferred on an interface or average data rate during the last n seconds). RTFM (Real-time Flow Monitoring) [1] and RMON (Remote Monitoring) [19] also use SNMP for transporting measured data. None of these solution offers any support for user-based IP accounting.

Commercial accounting systems, such as NetFlow [6], NARUS [11], or Juniper's J-Flow Accounting [8], lack the support for accounting on a per-user basis. The authors of [4] propose a method for accounting TCP connections on a per-user basis. Their solution is based on introducing an additional step in the TCP connection set-up phase for checking the authenticity of the user. If a TCP session is started all the traffic is reported to the user, who started the session. [20] proposes to use multiple IP addresses on a multi-user devices and use a distinct IP address for every user. This would allow traditional IP traffic accounting systems to be used for user-based IP accounting. NIPON [13] and [21] introduce an agent-based solution, where an agent is set up on a host with multiple users; this agent is designed to collect required traffic information directly on that host, but without having to change the operating system (kernel), so that it can also be used with closed source systems, like Solaris or Windows. The solution is based on capturing all traffic on the network interface to identify local traffic and to correlate it to local users. The drawback of this solution is the need to monitor all traffic on the link in case of shared links such as Fast Ethernet.

The UTAdragon project (Useful Traffic Accounting Dragon) [17] retrieves network data by collecting network and process information using the `/proc/net` interface. Data about open network connections and processes that use them are collected and recombined to create a table showing, which system user has consumed how much network traffic. The accounting data is stored in a MySQL Database, allowing further processing or aggregation. UserIPAcct (User IP Accounting) [16] is an extension to the Linux kernel originally developed in 1994. The development has stopped in 2000 and the latest beta version available addresses kernel 2.2. This system extends the Linux kernel in a way that it becomes able to attribute IP traffic to local system users. This code is not compatible with modern Linux kernels (*e.g.,* 2.6) and also does not support IPv6.

Comparing to existing work, LINUBIA proposes a user-based IP accounting system embedded into the latest generation of the Linux kernel (v2.6) and capable of performing accounting for IPv4 as well as IPv6 traffic. Moreover, LINUBIA reports measured

traffic separated on different transport protocols. An important difference to previous approaches is having not only a Linux kernel extension for user-based IP accounting, but a solution that can easily integrate into existing authentication and accounting systems by using standard protocols such as LDAP and Diameter. Havingthe accounting module embedded into the Linux kernel enables, besides traffic accounting, later extensions to perform IP traffic access control based on users or applications which generated the traffic. The solution proposed here follows an architecture close to the one proposed in [16].

3 Design

Based on the following use cases a summary of the main requirements for the new user-based IP accounting architecture, termed LINUBIA is derived. Based on these requirements that have been identified the design of the proposed solution is detailed.

3.1 Motivating Use Cases

Network Traffic Billing System
The first scenario deals with the case of a grid infrastructure spanning across a larger area on top of which customers may run their own grid applications. A grid user will typically install its applications on multiple nodes and these run typically with the user's privileges. The grid operator may use the user-based accounting module in order to split network costs (traffic created by grid applications is typically high) among all customers based on the amount of traffic they created.

Individual Load Monitoring and Abuse Detection
The second scenario addresses the case of an institution, for example a university, which offers its students the possibility to use the Web for research and communication purposes, but does not want them to excessively waste precious network bandwidth for sharing videos, filesharing, and the like. The system setup is done in a way that a student can log into one of many computers at the university with his personal credentials. The user account information is stored in a centralized LDAP directory, so a specific student has uses the same user ID (UID) in every system he logs into. A script can regularly copy usage information to a database server, where it is stored and accumulated with the traffic footprint of other users in order to detect possible anomalies in the traffic under investigation. The system administrator has the possibility to monitor network usage of students, independent of applications or the computer they use. With the help of this information he can detect and quantify abuses, suspend accounts of the respective users, or initiate further investigations.

Service Load Measuring
The third scenario handles the identification of applications, which generate abnormal traffic. For example, on a Linux server different services may be operational, some of them may not be using well-known ports (*e.g.,* a bit-torrent client, which constantly changes ports it is running on). On that router connecting this server to the Internet, the administrator can monitor how much traffic this server created, but he can only

identify applications based on port numbers. In case of applications that change these ports the use a user-based IP accounting module eases traffic monitoring for these type of applications.

3.2 Requirements

Based on these use cases above as well as a general observation of achieving a practical and implementable solution the following four key requirements for IP accounting LINUBIA have been identified:

- The IP accounting module shall account for IPv4 and IPv6 traffic information on a per-user bases operating the Linux operating system.
- The IP accounting module shall allow for application-based traffic accounting.
- An API interface shall be available for configuring the IP accounting module and retrieving accounted for data.
- The performance impact of the IP accounting module on the networking subsystem should be kept minimal.

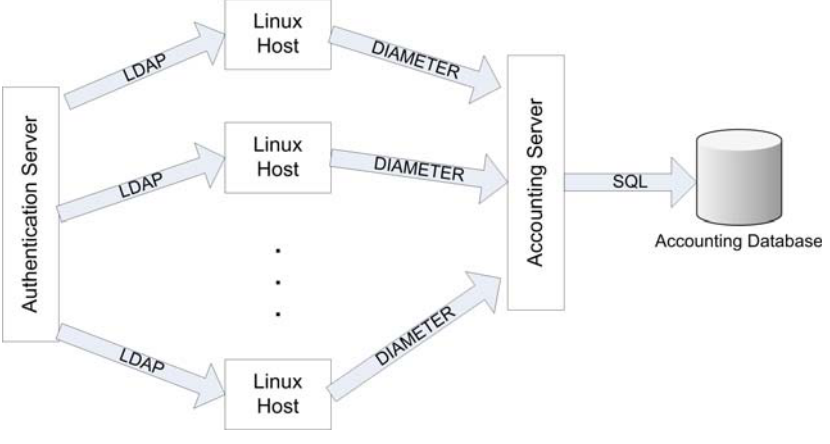

Fig. 2. Network Architecture

3.3 LINUBIA Architecture

The architecture of an enterprise network having LINUBIA running on the Linux end-hosts, consists of both a network architecture (cf. Fig. 2) that defines the network components required for LINUBIA and a end-host architecture (cf. Fig. 3) that defines the software components required within an end-system to support LINUBIA.

Two types of devices may be identified: regular Linux hosts, which may be used by users and the accounting domain infrastructure (consisting of an authentication server, a data aggregation server, and a storage server). Linux nodes use an authentication server for verifying user credentials. Whenever a user logs in to a Linux host all the processes started by the user will run with the global UID of the user. Each Linux host has LINUBIA enabled. Accounted for data is encapsulated in accounting records and it is

transported from each Linux host to an accounting server using the Diameter protocol. The accounting server further stores the accounting records in a central database. For supporting this an accounting client runs on each host, collects the data accounted by LINUBIA and sends it to an accounting server using the Diameter protocol.

3.3.1 End-Host Accounting Architecture

This section shall describe in detail the different components and their interactions required within an end-host in order to support user-based IP accounting. Regular operating systems do not offer a function to autonomously measure user-specific IP traffic. Therefore, a host needs to be modified in order to be able to perform such a task. Fig. 3 shows how this can be achieved by modifying the Linux operating system kernel, which resides between the networking hardware and applications in the user space. The kernel allows network applications to access the TCP/IP stack via the network socket interface; it contains routines to send outgoing IP packets to the network and deliver incoming packets to the destination applications. These routines and the kernel have to be extended in order to measure, store and export the desired accounting information associated with each accounting-relevant IP network operation. This is done by a kernel accounting extension that consists of a number of components which are added to the kernel.

The *information storage* component is responsible for the temporary storage of accounting information collected. Each incoming or outgoing packet triggers a lookup

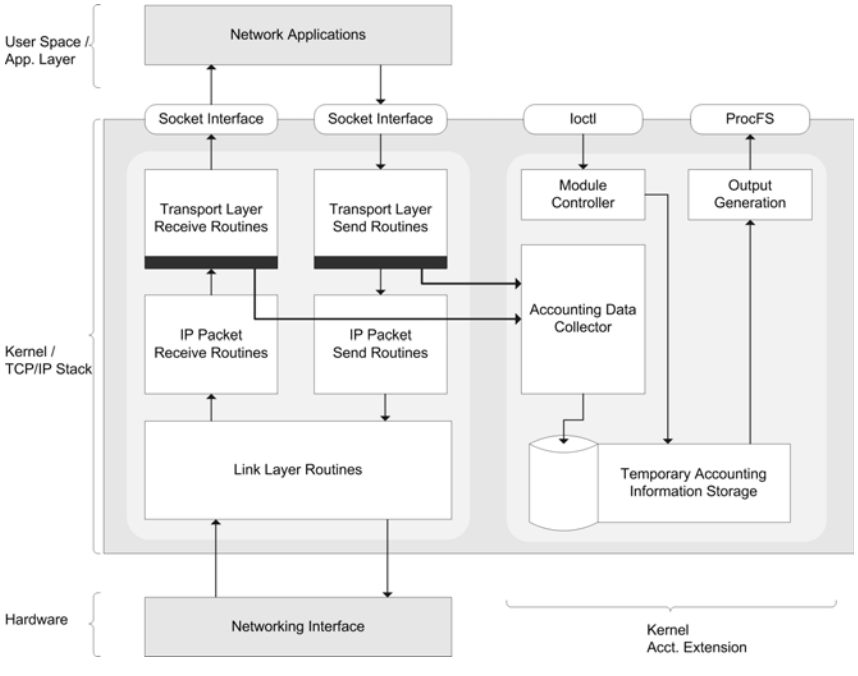

Fig. 3. End-Host Architecture

in this component for finding the record entry for the username responsible with the transfer thus the efficiency of the *information storage* component highly impacts the overall performance of the accounting module. The *data collector* component retrieves the necessary information from the IP networking subsystem and puts it in the storage component. The *output generation* component reformats the internal data before exporting it to user space via the proc filesystem (`procfs`). The *module controller* provides facilities to manage records stored, for example to reset all records of a specific user. It uses the `ioctl` interface.

This architecture is designed to extract and export user-specific IP accounting information from the kernel to user space for further processing. The data is stored temporarily in the main memory by the kernel module. Data aggregation and persistent storage are done outside the kernel. in order to keep low the load on the kernel.

3.3.2 Integrated View

Fig. 4 shows the integration of the host specific architecture into the network architecture. In addition to the kernel-based accounting architecture sketched in 3.3.1 two additional components are required for building accounting applications on top of LINBUIA. The first component is an accounting library that provides the API for querying and configuring the accounting module. It enables applications to access the kernel interfaces of the accounting extension.

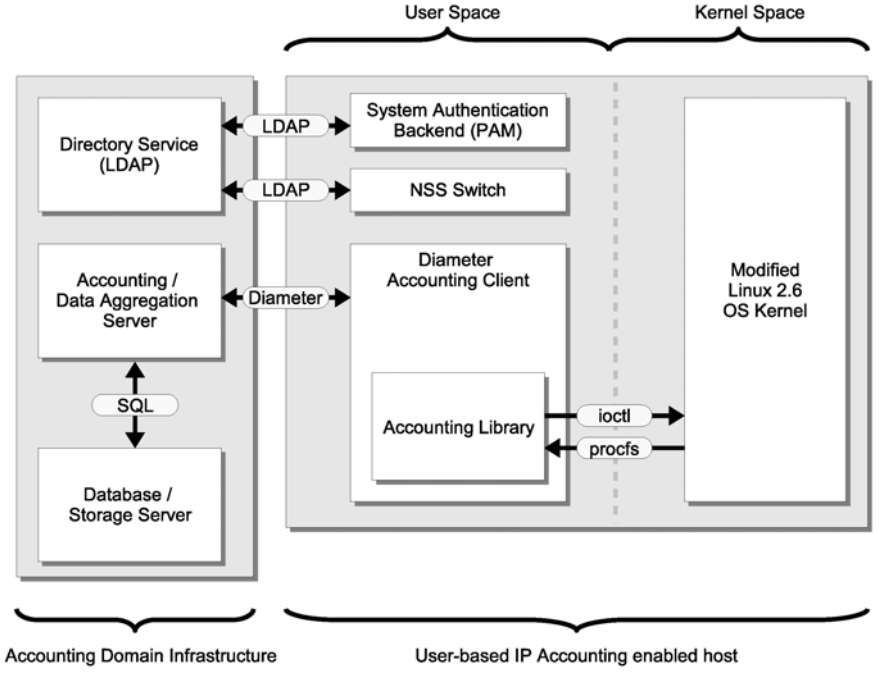

Fig. 4. Integrated View

The second component is a Diameter accounting client that uses this library to fetch the user-based IP accounting records from the kernel and sends them to a remote data aggregation server using the Diameter protocol. The aggregation server can evaluate and store the accounting data persistently, for example by using a separate database server.

A flexible system authentication back-end and Name Service Switch (NSS) configuration allows that a unique user account of a centralized user database (on a remote directory) can be used on any user host; the suggested interface being used for this is LDAP. The intention is that multiple hosts use the same user database and therefore the same UIDs for individual users, making users and associated accounting records uniquely identifiable across distinct hosts.

4 Implementation

The implementation of the host-based extension is based on the code layout of the useripacct project [16] and is entirely written in the C programming language [5]. Compared to the other investigated approaches, LINUBIA supports 64 bit counters, provides real-time traffic statistics and allows parallel accounting of IPv4 as well as IPv6. The accounting system was implemented for modern 2.6 series Linux kernels and supports both IPv4 and IPv6.

The information triplet to be extracted from each IP network operation consists of the IP packet size, the packet owner (user), and the network and transport protocols involved with the operation. Unfortunately, the required routines and protocol headers are distinct for IPv4 and IPv6, and for incoming traffic, the information cannot be retrieved at the IP layer, like it is the case for outgoing traffic. This required the embedding of the accounting module routines in the transport layer implementation. A shortcoming of this approach is a scatter of the LINUBIA code across several files in the Linux kernel network subsystem.

The *data collector* can extract the size of a packet from IP packet headers; the sum of the transferred IP packet sizes equals the IP traffic. The network and transport protocol types can be determined by identifying the kind of the network routine or by also inspecting the IP packet header. The user information can be determined by looking up the ownership properties of the network socket corresponding to a packet. As it is possible that IP packets are sent or received that have no associated local network socket, there are rare situations where traffic cannot be attributed to a regular user. This is handled by directing such accounting information to the record of a special user “nobody”.

The *information* storage component is implemented as a number of records that are connected in groups of doubly-linked lists within a hash table. Each record contains the UID as the primary identification attribute as well as the measured IP traffic values for different network and transport protocols. Users are dynamically added when they start using IP-based networking.

Upon request, the *output generation* component loops through these lists to create a table with all users and their traffic records which is exported to the `proc` file system. The user space library reads a special item in the `proc` filesystem that is exported by the kernel extension and contains the temporary accounting information. The library recreates the record structures so that they can be easily accessed by other applications,

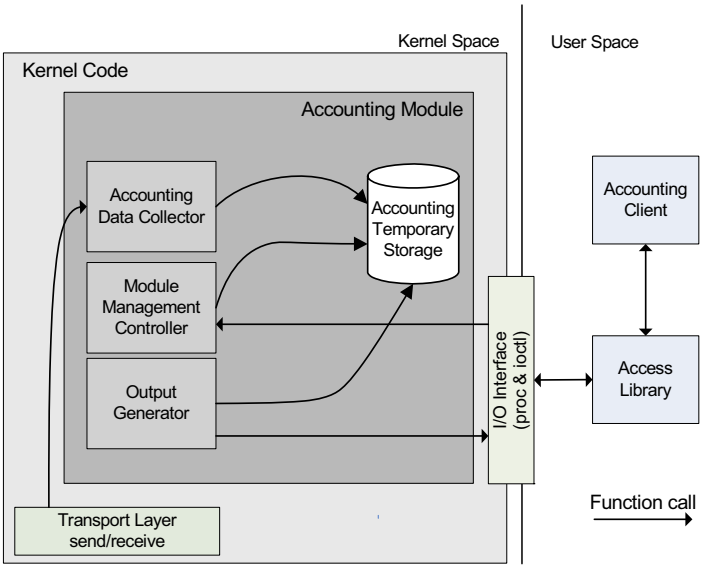

Fig. 5. Implementation Architecture

such as the accounting client. It also provides functions to send commands to the *module controller*, using the `ioctl` interface. The accounting client sends locally detected accounting records to the accounting server using the Diameter protocol. Within Diameter, records are structured as sets of (predefined) Attribute- Value Pairs (AVP). The sample accounting client and sample server communicate in regular intervals by using accounting sessions, where an accounting session contains current records for one user, as delivered by the accounting library. Besides the accounting Attribute-Value-Pairs (AVPs) proposed in [2], a set of parameters have been defined as shown in Tab. 1.

A patch containing the user-based IP accounting module for the 2.6.17 version of the Linux kernel may be found at *http://www.csg.uzh.ch/staff/morariu/linubia.*

Table 1. New AVPs for Linux User-Based IP Accounting

AVP Name	AVP Code	AVP Name	AVP Code
Linux-Input-IPV4-Octets	5001	Linux-Input-IPV4-TCP-Octets	5101
Linux-Output-IPV4-Octets	5002	Linux-Output-IPV4-TCP-Octets	5102
Linux-Input-IPV6-Octets	5003	Linux-Input-IPV4-UDP-Octets	5103
Linux-Output-IPV6-Octets	5004	Linux-Output-IPV4-UDP-Octets	5104
Linux-Input-TCP-Octets	5005	Linux-Input-IPV6-TCP-Octets	5105
Linux-Output-TCP-Octets	5006	Linux-Output-IPV6-TCP-Octets	5106
Linux-Input-UDP-Octets	5007	Linux-Input-IPV6-UDP-Octets	5107
Linux-Output-UDP-Octets	5008	Linux-Output-IPV6-UDP-Octets	5108

5 Evaluation

The evaluation of the user-based IP accounting module was performed both in terms of functional and performance evaluation. The tests have shown that the requirements described in Sect. 3.2 have been fully met. The set of experiments that have been performed in order to test the functionality, accuracy and performance of the accounting module used a network set-up as the one described in Fig. 6. The testing environment consists of two hosts that are connected in a LAN by a Fast Ethernet switch as seen in the figure. Both hosts run a Linux 2.6 operating system and use IPv4 as well as IPv6. Both hosts have Fast Ethernet network adapters. All performance tests have been per-formed in a laboratory environment. For testing the functionality and robustness of the module LINUBAIA was installed on an Ubuntu desktop machine and used in a production environment.

For testing the accuracy of the accounting module several tests have been performed in which TCP, UDP, and ICMP incoming and outgoing IPv4 and IPv6 traffic was generated and accounted for. The experiments have shown that the accounting module correctly accounts for IP traffic. During experiments it was observed that some traffic cannot be mapped to any user (such as scanning traffic or incoming ICMP messages). Such traffic is accounted for the system user by the accounting module. Another observation concerns ICMP traffic that appears to be exclusively mapped to the system user and not to the user who actually sent the message. The reason for this is that raw socket operations are considered critical and only possible for user root, also for security reasons (a regular user can only execute the ping program because it has the SUID-bit set, thus being executed under root context).

Tab. 2 shows the results of a first test consisting of a 256 MB file transfer over a Fast Ethernet link with and without LINUBIA using IPv4 and IPv6. The purpose of this test was to identify the impact of accounting on the performance of the Linux network subsystem. As the table shows there is only a small impact (0.83 for IPv4 and 0.41 for IPv6) on performance observed when running with LINUBIA enabled.

In Tab. 3 observed and estimated maximum throughput on a Linux box with and without LINUBIA are shown. For estimating the maximum throughput the Iperf [7] tool was used. The test with Iperf affirms that the measuring results are correct. Although

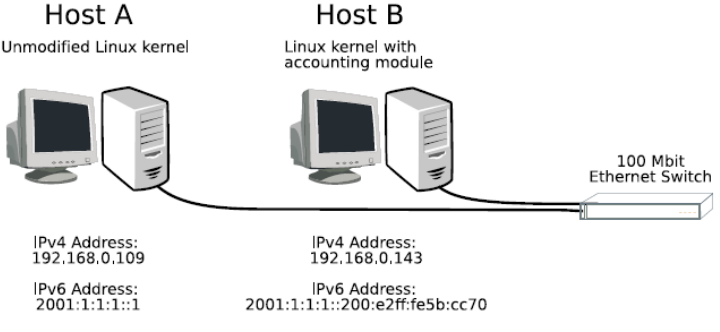

Fig. 6. Testing environment

Table 2. Average time for a 256 MB file transfer over a Fast Ethernet connection with and without the user-based IP accounting enabled (average numbers of 20 runs)

	Unmodified IPv4	Accounting IPv4	Unmodified IPv6	Accounting IPv6
Average time	21.815 s	21.998 s	22.102 s	22.193 s
Std. deviation	0.062 s	0.208 s	0.010 s	0.204 s

Table 3. Average maximum throughput observed and calculated by Iperf over an Fast Ethernec connection, with and without the user-based IP accounting module enabled.

	Unmod. IPv4	Acct. IPv4	Rel. diff. (%)	Unmod. IPv6	Acct IPv6	Rel. diff. (%)
Manual (Mbps)	93.880	93.099	0.839	92.661	92.281	0.412
Iperf (Mbps)	94.080	91.700	2.595	92.880	92.870	0.012

the values are not totally equal, the dimensions are the same and the performance loss is marginal.

During the evaluation phase of LINUBIA the architecture and its implementation have been tested to check the functionality they provide and the performance impact on the Linux kernel network subsystem. These tests have shown that LINUBIA delivers the required accounting results, especially a per-user network activities result on a multi-user operating system, while having a small impact on the performance of the endsystem under investigation.

6 Conclusions and Future Work

This paper demonstrates by a design and prototypical implementation that a userbased IP accounting approach is technically possible on modern Linux (2.6 series) operating systems. Additionally, it can be used also in the same version with the upcoming IPv6 network protocol and it can be integrated into an existing accounting infrastructure, such as Diameter. On one hand, users are not supposed to have only one computer device of their own (not to mention sharing one device with other users), but rather to have several devices for different purposes. On the other hand, the more computers become commodities for daily life and will be used by different people (producing networking-related and other costs), the more important it becomes to establish accounting systems, which offer a clear and secure user identification on the end-device and will probably have an integrated character. The current implementation shows a clear proof of concept. Compared to traditional device-based accounting mechanisms, a user-based approach allows the mapping of network services usage not only to a device, but more specific, to the user which consumed those services.

Improvements are possible, *e.g.,* with the storage component, which can be done with a smaller memory footprint and also more efficiently by utilizing advanced data

structures that will help to optimize access times. Another interesting issue determines the linkage of the networking subsystem to the socket interface, which also implies a link to the process management of the operating system. An advanced accounting module can offer IP accounting not only per user, but also per process. This allows for the identification, the management, or schedulability of processes not only by their CPU usage or memory consumption, but also by their network resource consumption. Finally, this leads to the creation of network filters or firewalls that allow for or deny network access to specific applications or users running on a host, instead of only allowing or denying specific services. The current LINUBIA implementation treats all traffic the same, thus producing an overall network consumption report for each user. An interesting improvement would be separated accounting for different services (differentiated based on DSCP number or destination Autonomous System).

Acknowledgment

This work has been performed partially in the framework of the EU IST Project EC-GIN (FP6-2006-IST-045256) as well as of the EU IST NoE EMANICS (FP6-2004-IST-026854).

References

1. Brownlee, N., Mills, C., Ruth, G.: Traffic Flow Measurement: Architecture; RFC 2722 (October 1999)
2. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter Base Protocol; RFC 3588 (September 2003)
3. Harrington, D., Presuhn, R., Wijnen, B.: An Architecture for Describing SNMP Management Frameworks; RFC 2271 (January 1998)
4. Edell, R.J., McKeown, N., Varaiya, P.P.: Billing Users and Pricing for TCP. IEEE Journal on Selected Areas in Communications 13(7) (September 1995)
5. Feier, M.: Design and Prototypical Implementation of a User-based IP Accounting Module for Linux, Diploma Thesis, University of Zürich, Switzerland (February 2007)
6. Flexible NetFlow Homepage (May 2007), `http://www.cisco.com/en/US/products/ps6601/products_data_sheet0900aecd804b590b.html`
7. Iperf Homepage: (May 2007), `http://dast.nlanr.net/Projects/Iperf/`
8. Juniper Homepage (May 2007), `http://www.juniper.net/`
9. Koukal, M., Bestak, R.: Architecture of IP Multimedia Subsystem. In: 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, Zadar, Croatia (June 2006)
10. de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., Spence, D.: Generic AAA Architecture; RFC 2903 (August 2000)
11. Narus Homepage (May 2007), `http://www.narus.com/`
12. Neuman, B.C., Ts'o, T.: Kerberos: An Authentication Service for Computer Networks. IEEE Communications 32(9), 33–38 (1994)
13. NIPON: Nutzerbasiertes IP Accounting (May 2007), `http://www.icsy.de/forschung/nipon/`
14. Rigney, C., Willens, S., Rubens, A., Simpson, W.: Remote Authentication Dia. In: User Service (RADIUS); RFC2865 (June 2000)

15. Sermersheim, J. (ed.): Lightweight Directory Access Protocol (LDAP): The Protocol; RFC 4511 (June 2006)
16. UserIPacct Homepage (May 2007), http://ramses.smeyers.be/homepage/useripacct/
17. UTA Dragon Homepage (May 2007), http://www.crash-override.net/utadragon.html
18. Vinton, G. (ed.): Guidelines for Internet Measurement Activities; RFC 1262, (October 1991)
19. Waldbusser, S.: Remote Network Monitoring Management Information Base; RFC 2819 (May 2000)
20. Zhang, G., Reuther, B.: A Model for User Based Traffic Accounting. In: 31st EUROMICRO Conference on Software Engineering and Advanced Applications, Porto, Portugal (August 30–September 3, 2005)
21. Zhang, G., Reuther, B., Mueller, P.: Distributed Agent Method for User Based IP Accounting. In: 7th CaberNet Radicals Workshop, Bertinoro, Forlí, Italy (October 13-16, 2002)